Conditional Differential Cryptanalysis of Grain-128a

Michael Lehmann and Willi Meier

FHNW, Switzerland

Abstract. Grain-128a is a new version of the stream cipher Grain-128. To analyse the security of the cipher, we study the monomial structure and use high order differential attacks on both the new and old versions. The comparison of symbolic expressions suggests that Grain-128a is immune against dynamic cube attacks. Additionally, we find that it is also immune against differential attacks as the best attack we could find results in a bias at round 189 out of 256.

Keywords: stream ciphers, Grain-128, Grain-128a, conditional differential cryptanalysis.

1 Introduction

Grain is a family of lightweight stream ciphers that share the property of a very small hardware implementation. As any modern stream cipher, Grain allows for public initial vectors so that the initial state for keystream generation is produced by an initialization mechanism that depends on the secret key and on the initial vector. There are two versions, Grain v1 [9] with a 80-bit key, and Grain-128a [1] with a 128-bit key. The latter has built-in support for optional authentication. Grain v1 is a finalist in the eSTREAM portfolio of hardware oriented stream ciphers. Grain-128a is modelled on its predecessor Grain-128 [8], but uses slightly different non-linear functions with the aim to strengthen it against known attacks on Grain-128.

Grain v1 and Grain-128a share a very similar structure based on non-linear feedback shift registers (NFSR). In [10], conditional differential cryptanalysis, first introduced in [4], has been applied to such constructions. The idea is to control the propagation of differences by imposing conditions on the public variables, i.e. the initial vector (IV), of the cipher. Depending whether these conditions involve secret variables or not, key-recovery or distinguishing attacks can be mounted. The technique extends to higher order differential cryptanalysis. A different but related concept is the dynamic cube attack presented in [7]. Because of the higher complexity of the update functions of Grain v1, the attacks are not applicable to this cipher.

The aim of this paper is to compare the security of Grain-128a with that of Grain-128 with regard to higher order differential attacks, including conditional differential cryptanalysis. By studying the monomial structure in the initialisation mechanism it is argued that dynamic cube attacks will not be effective

J. Pieprzyk, A.-R. Sadeghi, and M. Manulis (Eds.): CANS 2012, LNCS 7712, pp. 1–11, 2012.

against Grain-128a as was the case for Grain-128. This is the first analysis of the security of Grain-128a in a chosen IV scenario.

The paper is organized as follows. Section 2 gives a brief summary of the cube attack as well as the dynamic cube attack and recalls the idea of conditional differential analysis. Section 3 gives a concise overview of the design of Grain-128a and the changes made with respect to its predecessor Grain-128. Section 4 provides a comparison of Grain-128a with its predecessor describing the impacts of the improvements made to the cipher. It shows the results achieved with higher order differential attacks in general and the conditional differential analysis in particular. In order to make a statement concerning the security of the cipher, these results are again compared to results on Grain-128.

2 Background

This section briefly depicts the two different variants of the cube attack, namely the static and the dynamic variants. The section is a summary of [6], [7] and [5], respectively. Initial work related to cube attacks is also [13]. Section 2.3 recalls relevant facts on conditional differential analysis in a concise manner. For more detailed information we refer to [10].

2.1 Cube Attack

For most stream ciphers, an output bit can be described as a master polynomial $p(k_1, \ldots, k_n, v_1, \ldots, v_m)$ over $\mathbb{F}_2$. Such a polynomial can be split and written as a sum of two polynomials:

$$p(k_1, \ldots, k_n, v_1, \ldots, v_m) = t_I * p_{S(I)} + q(k_1, \ldots, k_n, v_1, \ldots, v_m)$$

Where:

- t_I is called maxterm and is a product of certain IV bits, for example $v_1 v_2 v_5$
- $p_{S(I)}$ is called superpoly. It does not contain any variables of t_I
- $q(k_1, \ldots, k_n, v_1, \ldots, v_m)$ is the remainder polynomial. The summands of this polynomial miss at least one of the variables of t_I.

The maxterm t_I is defined through a subset of indices I (a so-called cube). In order to get the super polynomial $p_{S(I)}$, one assigns all possible values to the variables contained in I, evaluates the master polynomial and sums up the results. In this way, every summand missing k (k$\geq$1) variables of the maxterm t_I is added exactly 2^k times and is therefore eventually eliminated with the modular reduction. The super polynomial $p_{S(I)}$, however, is part of the evaluated master polynomial if and only if all variables of t_I have value one. All other variables whose indices are not contained in I are assigned a certain value, usually zero. The idea of cube attacks is to find enough maxterms t_I whose super polynomial is linear and not a constant. This enables to recover the key through solving a system of linear equations.

2.2 Dynamic Cube Attack

The Dynamic Cube Attack is very similar to the static version. The difference is that certain variables which are not part of t_I are assigned a function of public and private variables instead of a constant value. Those functions are chosen in a way that the symbolic expressions of certain variables are simplified. The idea is to rewrite a polynomial P with three polynomials:

$$P = P_1 * P_2 + P_3$$

In order to simplify the polynomial P, one sets a linear term of P_1 in such a way that the whole polynomial P_1 is zero. This eliminates P_2 and simplifies P to P_3. An example is given in [7]. The attack was further improved and published in [5]. This attack breaks the full version of Grain-128 with a complexity of about 2^{90} and memory usage of 2^{63} bit.

2.3 Conditional Differential Analysis

The notion of a conditional differential characteristic has been introduced in [4] to improve differential attacks against DES. Similar ideas are used to accelerate the differential collision search in hash function cryptanalysis [14,15]. In [10,11] the principle has been applied to NFSR-based constructions and extended to higher order differential attacks.

The Basic Idea for NFSR-based Constructions. Let us consider the case of a synchronous stream cipher taking as its input an initial value (IV) and a key. We assume a scenario where the attacker can observe the keystream for many chosen IVs under the same secret key. The basic idea of conditional differential cryptanalysis is to control the propagation of a difference in the IV through the first few rounds of the initialization process. This is done by imposing specific conditions on certain bits of the IV. From these conditions, a sample of IV pairs is derived and experimentally tested for a bias in the resulting keystream differences. Conditions might be also imposed on the key which defines classes of weak keys.

An example for Grain-128a . Consider a difference in bit 69 of the IV. The difference does not affect the rounds 0 to 8. Then, at round 9, the value of the feedback is computed as $x_{17}k_{21}+x_{22}x_{29}+x_{51}k_{104}+x_{69}x_{88}+k_{11}+k_{21}k_{104}+k_{24}+k_{45}+k_{54}+k_{73}+k_{82}+k_{98}+1$, that is, the difference eventually propagates to s_{9+128} and b_{9+128}. Imposing the condition $x_{88} = 0$ we can prevent the difference from propagating. Similarly at round 27, h is computed as $x_{35}k_{39} + x_{40}x_{47} + x_{69}k_{122} + x_{87} + k_{29} + k_{39}k_{122} + k_{42} + k_{63} + k_{72} + k_{91} + k_{100} + k_{116} + 1$ and the propagation can be prevented by the condition $k_{122} = 0$. Testing a sample of 2^{16} randomly chosen IV pairs separated by a difference in bit 69, a significant bias can be detected at round 140. However, if the IVs satisfy the conditions $x_{88} = 0$ and the key satisfies $k_{122} = 0$, biases can be detected up to round 159.

Extension to Higher Orders. If the keystream bits are modeled as Boolean functions of the form $f : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}$, a first order attack evaluates

$$\Delta_v f(k,x) := f(k,x) + f(k,x+v)$$

for a fixed difference $v \in \{0,1\}^n$ many chosen $x \in \{0,1\}^n$. In [12], $\Delta_v f$ is called a first order derivative of f with respect to v. More generally, if $V \subset \{0,1\}^n$ is a linear subspace of dimension d,

$$\Delta_V f(k,x) := \sum_{v \in V} f(k,x+v)$$

is called the derivative of f with respect to V. The principle of conditional differential cryptanalysis extends to higher orders as follows. If $\{a_1, \ldots, a_d\}$ is a basis of V, conditions are derived for each b_i as a first order difference and merged to a total set of differences from which the sample is derived.

Notation and Terminology. In this paper, the subspaces V will be always generated by IVs of Hamming weight one and we write v_i, $0 \le i \le 95$ for the IV with a 1 at position i and 0s otherwise. Conditional differential cryptanalysis can be seen as a refinement of cube testers introduced in [3].

3 Grain-128a

In our analysis, the authentication mechanism will be ignored. Fig. 1 depicts an overview of the building blocks of the output generator, which is constructed using three main building blocks, namely an LFSR, an NFSR and a pre-output function. We denote by $s_i, s_{i+1}, \ldots, s_{i+127}$ the contents of the LFSR. Similarly, the content of the NFSR is denoted by $b_i, b_{i+1}, \ldots, b_{i+127}$. Together, the 256 memory elements in the two shift registers represent the state of the output generator.

The primitive feedback polynomial of the LFSR, denoted $f(x)$, is defined as

$$f(x) = 1 + x^{32} + x^{47} + x^{58} + x^{90} + x^{121} + x^{128}.$$

We also recall the corresponding update function of the LFSR as

$$s_{i+128} = s_i + s_{i+7} + s_{i+38} + s_{i+70} + s_{i+81} + s_{i+96}.$$

The nonlinear feedback polynomial of the NFSR, $g(x)$, is defined as (changes to the predecessor Grain-128 are in boldface)

$$\begin{aligned} g(x) = {} & 1 + x^{32} + x^{37} + x^{72} + x^{102} + x^{128} + x^{44}x^{60} \\ & + x^{61}x^{125} + x^{63}x^{67} + x^{69}x^{101} \\ & + x^{80}x^{88} + x^{110}x^{111} + x^{115}x^{117} \\ & + \boldsymbol{x^{46}x^{50}x^{58}} + \boldsymbol{x^{103}x^{104}x^{106}} + \boldsymbol{x^{33}x^{35}x^{36}x^{40}}. \end{aligned}$$

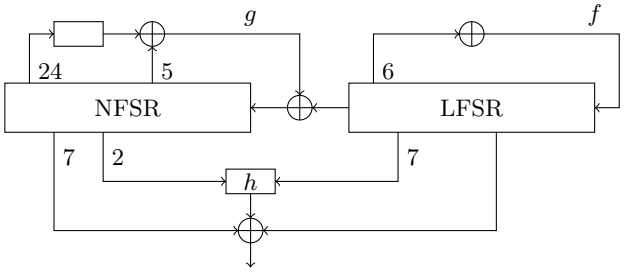

Fig. 1. An overview of the output generator

Again, recall the rule for updating the NFSR, with changes to Grain-128 in boldface.

$$\begin{aligned} b_{i+128} = {} & s_i + b_i + b_{i+26} + b_{i+56} + b_{i+91} + b_{i+96} \\ & + b_{i+3}b_{i+67} + b_{i+11}b_{i+13} + b_{i+17}b_{i+18} \\ & + b_{i+27}b_{i+59} + b_{i+40}b_{i+48} + b_{i+61}b_{i+65} \\ & + b_{i+68}b_{i+84} + \boldsymbol{b_{i+88}b_{i+92}b_{i+93}b_{i+95}} \\ & + \boldsymbol{b_{i+22}b_{i+24}b_{i+25}} + \boldsymbol{b_{i+70}b_{i+78}b_{i+82}}. \end{aligned}$$

Note that the update rule contains the bit s_i which is not part of the feedback polynomial and is output from the LFSR, thus masking the input to the NFSR.

Nine state variables are taken as input to a Boolean function, $h(x)$: two bits come from the NFSR and seven from the LFSR. This function is defined as

$$h(x) = x_0x_1 + x_2x_3 + x_4x_5 + x_6x_7 + x_0x_4x_8$$

where the variables $x_0, \ldots, x_8$ correspond to, respectively, the state variables b_{i+12}, s_{i+8}, s_{i+13}, s_{i+20}, b_{i+95}, s_{i+42}, s_{i+60}, s_{i+79} and s_{i+94} (or s_{i+95} for Grain-128, respectively). The pre-output function is defined as

$$y_i = h(x) + s_{i+93} + \sum_{j \in \mathcal{A}} b_{i+j},$$

where $\mathcal{A} = \{2, 15, 36, 45, 64, 73, 89\}$.

Before keystream is generated the cipher must be initialized with the key and the IV. Denote the bits of the key as $k_i,\ 0 \leq i \leq 127$ and the IV bits $IV_i,\ 0 \leq i \leq 95$. The initialisation of the key and IV is done as follows. The 128 NFSR elements are loaded with the key bits, $b_i = k_i,\ 0 \leq i \leq 127$, and the first 96 LFSR elements are loaded with the IV bits, $s_i = IV_i,\ 0 \leq i \leq 95$. The last 32 bits of the LFSR are filled with ones and a zero, $s_i = 1,\ 96 \leq i \leq 126$, $s_{127} = 0$. Then, the cipher is clocked 256 times without producing any keystream. Instead, the output function is fed back and xored with the input, both to the LFSR and to the NFSR, see Fig. 2.

In the mode without authentication, all output bits are used directly as keystream. This mode of operation is the same as in Grain-128.

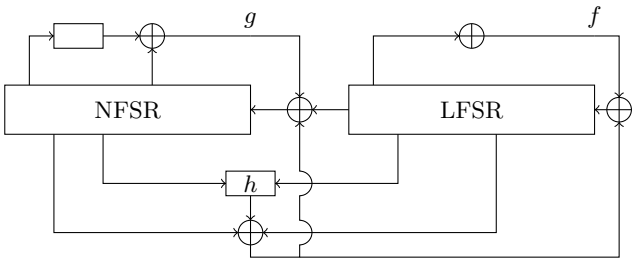

Fig. 2. The state initialization

4 Findings

The NFSR-update used by Grain-128 is merely of order two, whereas the one used by Grain-128a is of order four and contains two extra monomials of order three. Consequently, the symbolic expressions of Grain-128a grow faster. Fig. 3 depicts this fact using the data we collected.

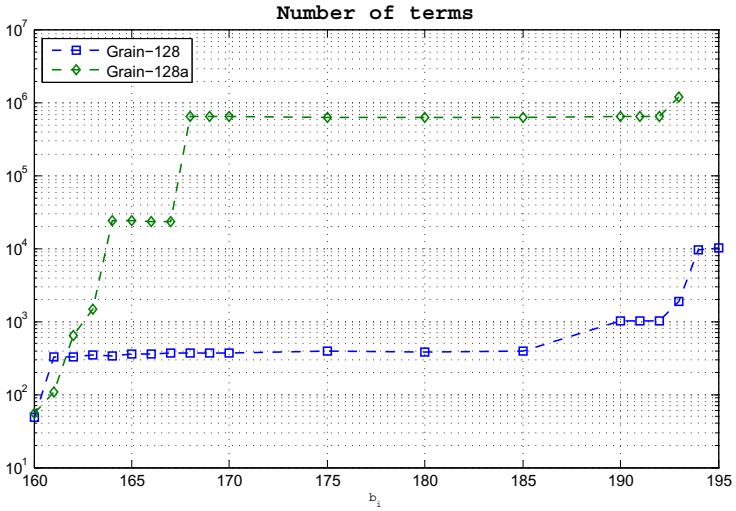

Fig. 3. Number of terms

To illustrate the difference relating to the order of the symbolic expressions, i.e. the order of monomials in variables b_i and s_i, and the number of terms of maximum order, Table 1 shows the order and number of terms of some b_i of Grain-128 and Grain-128a, respectively. Note that the two columns on the left both describe the number of monomials of order greater or equal the maximum order in the symbolic expression of Grain-128.

Table 1. Order of the symbolic expressions of the b_i for Grain-128 vs Grain-128a

	Order of symbolic expressions		**#terms $\geq$ max. order Grain-128**	
	Grain-128	**Grain-128a**	**Grain-128**	**Grain-128a**
b_{160}	3	4	1	7
b_{161}	6	7	7	4
b_{162}	5	8	35	153
b_{163}	5	9	35	768
b_{164}	5	11	35	20786
b_{165}	5	11	35	20757
b_{166}	5	11	35	20229
b_{167}	5	11	35	19701
b_{168}	5	13	35	597807
b_{169}	5	13	35	597807
b_{170}	5	13	35	597507
b_{175}	5	13	35	583133
b_{180}	5	13	28	577368
b_{185}	5	13	28	570672
b_{190}	5	13	28	583829
b_{191}	5	13	28	583514
b_{192}	5	13	28	583514
b_{193}	6	14	21	859311
b_{194}	9	*out of memory*	89	*out of memory*
b_{195}	8	*out of memory*	466	*out of memory*

Table 1 shows that after 36 rounds (b_{164}), the symbolic expression of Grain-128a is of order 11 as opposed to 5 in the case of Grain-128, but even more striking is the difference in the number of terms. After the same 36 rounds, the number of terms of order 5 or greater is 35 for Grain-128 compared to 20786 for Grain-128a, which is over 590 times more.

An interesting fact is that the order of the expressions of Grain-128 stays the same for rounds 34 - 64. The reason is that the term of highest order is part of the pre-output function and the LFSR is filled with ones during the initialisation. The seven monomials of maximum order 6 in Grain-128's b_{161} are the following:

$$\boldsymbol{s_{95}} b_{45} b_{12} b_{95} (b_3 b_{67} + b_{11} b_{13} + b_{17} b_{18} + b_{27} b_{59} + b_{40} b_{48} + b_{61} b_{65} + b_{68} b_{84})$$

For the next 31 rounds, the bit s_{i+95} will have the value one, hence reducing the order to 5 and keeping it on that same level during those rounds. The growth of the number of terms of maximum order has the same cause. As the terms of maximum order are determined through those of the pre-output function and the monomial of order 3 is only of order 2 during those 31 rounds, there are five monomials of maximum order 2 in the pre-output function, as opposed to one of maximum order 3 for b_{161}.

Grain-128a's monomial with maximum order lies in the NFSR update:

$$b_{i+88}b_{i+92}b_{i+93}b_{i+95}$$

The smallest index reaches 128 exactly four rounds later than the other indices, which results in the stall of the order from b_{164} up to b_{168}. After that, the order of this term is determined by the sub-terms, i.e. the monomials in the expanded expressions of the variables, of order 3 and 4. No more indices reach 128 before b_{193}. This b_{193}, however, contains the term $b_{153}b_{157}b_{158}b_{160}$, and b_{160}, in turn, contains the bit b_{128} (b_{i+96} in the NFSR update) which results in the observed order 14.

4.1 Higher Order Differential Analysis on Grain-128 and Grain-128a

In order to investigate the impacts of the higher order and the higher number of terms of the symbolic expressions, a higher order differential attack is conducted with random cubes of different dimensions. For each dimension, sums over 100 random cubes are calculated and in each case the last round with a significant bias is identified. The number of random IVs used per cube is $2^{12} = 4096$ and the significance level for the frequency test is 0.001. The result, i.e. the last rounds with a bias for each dimension, is shown as a Boxplot in the following Fig. 4. Note that the upper boxes are the results of the analysis of Grain-128, whereas the lower boxes correspond to Grain-128a.

Not only are the sums biased up to many more rounds for Grain-128 than for Grain-128a, but the dimension of the cube also has a much bigger influence.

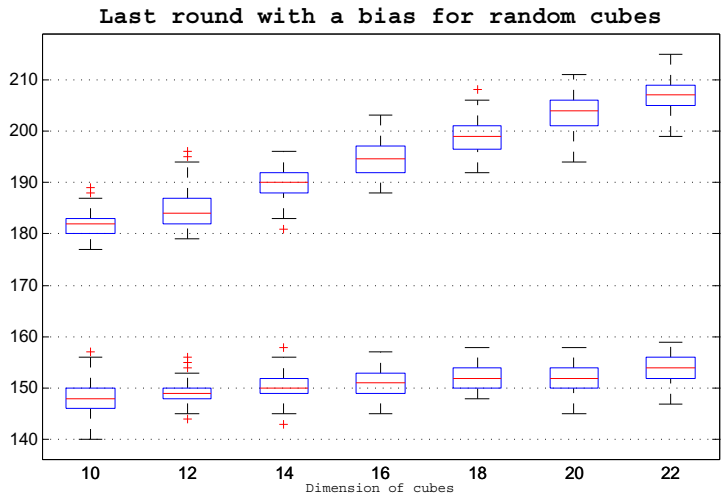

Fig. 4. Boxplot of last bias for random cubes

Cubes constructed by looking at the conditions of the cube bits and combining those with similar conditions, however, yield better results. For example, a random cube of dimension 20 results in a bias approximately up to round 152 without conditions (cf. Fig. 4) or up to round 160 with conditions, whereas the constructed cube of the same dimension results in a bias up until round 167.

4.2 Conditional Differential Analysis on Grain-128 and Grain-128a

To find the mentioned constructed cubes, we first symbolically sum over cubes of dimension one and examine the obtained symbolic expression of this sum over the cipher output z_i, thus getting the conditions of all public variables. The next step is to look for indices with the same conditions, as they seem to usually lead to good cubes. The reason is that the conditions generally do not occur at the same round and therefore one condition prevents that derivatives propagate at different rounds. Calculating the sum over v_{20}, for example, one finds the condition v_{27} at round 7 (z_7), whereas when calculating the sum over v_{34}, one finds the same condition at round 14 (z_{14}).

The following Table 2 lists the best results, i.e. the number of rounds attacked, on Grain-128 and Grain-128a for various cube dimensions, using 2048 random IVs and a significance level of 0.01:

Table 2. Best results for various cube dimensions on Grain-128 and Grain-128a

Cube dimension	12	16	20	24	28	33
Grain-128	207	215	219	225	231	236
Grain-128a	164	165	167	172	175	177

The best cube is of dimension 33 and results in a bias of approximately 0.463 at round 177. The public variables used as cube are the following:

$$v_1, v_2, v_3, v_{20}, v_{21}, v_{22}, v_{23}, v_{24}, v_{25}, v_{26}, v_{34}, v_{35}, v_{36}, v_{37}, v_{48}, v_{49}, v_{50},$$
$$v_{51}, v_{52}, v_{53}, v_{54}, v_{63}, v_{64}, v_{65}, v_{66}, v_{67}, v_{68}, v_{69}, v_{77}, v_{78}, v_{79}, v_{80}, v_{95}$$

If we apply the same cube to Grain-128, we find a bias of approximately 0.469 at round 236. In [2], the best cube found is of dimension 40 and results in a bias up to round 237. Furthermore, we find that the initialisation of Grain-128a is clearly much better as we find the last bias 59 rounds earlier with the same cube.

The results presented so far are achieved by imposing conditions only in the public variables. If we consider conditions in the private variables of Grain-128a as well, we get even better results in much less computing time. Evaluating the sum over the cube

$$v_{64}, v_{65}, v_{66}, v_{67}, v_{68}, v_{69}$$

of dimension 6, we find a significant bias at round 189. The conditions, i.e. the public and private variables set to a certain value, imposed here are the following:

$$v_{57}, v_{58}, v_{59}, v_{60}, v_{61}, v_{62}, v_{71}, v_{72}, v_{73}, v_{74}, v_{75}, v_{76}, v_{83}, v_{84}, v_{85}, v_{86}, v_{87}, v_{88}$$

$$k_{117}, k_{118}, k_{119}, k_{120}, k_{121}, k_{122}$$

Note that in order to attack the mentioned 189 rounds, all conditions have to be set to zero. This is the best attack we could find using conditional differential analysis. Based on the data collected and described in this section, we do not expect significantly better results to be found using this approach.

5 Conclusion

We evaluated the security of the stream cipher Grain-128a by comparing the monomial structure to its predecessor Grain-128. The significantly higher order of the symbolic expressions and the significantly higher number of monomials suggests that the dynamic cube attack, which breaks the predecessor Grain-128, is not applicable to Grain-128a. Additionally, we evaluated the security with respect to higher order differential attacks including conditional differential cryptanalysis. We used an automatic approach to find and analyse the conditions in terms of polynomial ideals. The attack of order 6 for 189 out of 256 rounds is the best attack we could find. Imposing conditions in only the public variables, the best attack found was of order 33 and attacked round 177. Consequently, Grain128a seems to have a comfortable security margin with respect to the approaches described in this paper.

Acknowledgements. We thank the reviewers of CANS 2012 for their helpful comments and Simon Knellwolf for his support and the contribution of the results with conditions in the private variables.

References

1. Ågren, M., Hell, M., Johansson, T., Meier, W.: Grain-128a: A New Version of Grain-128 with Optional Authentication. IJWMC 5(1), 48–59 (2011)
2. Aumasson, J.P., Dinur, I., Henzen, L., Meier, W., Shamir, A.: Efficient FPGA Implementations of High-Dimensional Cube Testers on the Stream Cipher Grain-128. Cryptology ePrint Archive, Report 2009/218 (2009), http://eprint.iacr.org/
3. Aumasson, J.-P., Dinur, I., Meier, W., Shamir, A.: Cube Testers and Key Recovery Attacks on Reduced-Round MD6 and Trivium. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 1–22. Springer, Heidelberg (2009)
4. Ben-Aroya, I., Biham, E.: Differential Cryptanalysis of Lucifer. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 187–199. Springer, Heidelberg (1994)
5. Dinur, I., Güneysu, T., Paar, C., Shamir, A., Zimmermann, R.: An Experimentally Verified Attack on Full Grain-128 Using Dedicated Reconfigurable Hardware. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 327–343. Springer, Heidelberg (2011)

6. Dinur, I., Shamir, A.: Cube Attacks on Tweakable Black Box Polynomials. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 278–299. Springer, Heidelberg (2009)
7. Dinur, I., Shamir, A.: Breaking Grain-128 with Dynamic Cube Attacks. Cryptology ePrint Archive, Report 2010/570 (2010)
8. Hell, M., Johansson, T., Maximov, A., Meier, W.: A Stream Cipher Proposal: Grain-128. In: ISIT, pp. 1614–1618 (2006)
9. Hell, M., Johansson, T., Meier, W.: Grain: A Stream Cipher for Constrained Environments. IJWMC 2(1), 86–93 (2007)
10. Knellwolf, S., Meier, W., Naya-Plasencia, M.: Conditional Differential Cryptanalysis of NLFSR-Based Cryptosystems. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 130–145. Springer, Heidelberg (2010)
11. Knellwolf, S., Meier, W., Naya-Plasencia, M.: Conditional Differential Cryptanalysis of Trivium and KATAN. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 200–212. Springer, Heidelberg (2012)
12. Lai, X.: Higher order derivatives and differential cryptanalysis. In: Blahut, R.E., Costello, D.J., Maurer, U., Mittelholzer, T. (eds.) Communicationis and Cryptography: Two Sides of one Tapestry, pp. 227–233. Kluwer Academic Publishers (1994)
13. Vielhaber, M.: Breaking one.fivium by aida an algebraic iv differential attack. IACR Cryptology ePrint Archive 2007, 413 (2007)
14. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
15. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)